

Travail 3 Partie 1

Pondération: 16% au total

Remise : jeudi 18 avril avant minuit (via Léa)
Enlevez le .db, le ipch et les dossiers debug

Le but du travail est principalement de mettre en pratique la notion de **polymorphisme**.

Le travail peut se faire en équipe de. Vous devez m'indiquer le nom des coéquipiers aujourd'hui.


Exigences

- Chaque classe doit être programmée à l'intérieur de ses propres fichiers (.h et .cpp)
- Vous devrez faire des classes C++ standard.
- **Utilisez vos objets le plus souvent possible et non les valeurs affichées à l'écran.**
- Respectez les standards de programmation et les bonnes techniques de programmation.
- Respectez les principes de la programmation objet.

Description du travail

Tel que vous avez discuté dans le cours d'analyse, nous allons faire une application qui manipule différents types de véhicule. Chacun d'eux devra apparaître au bon endroit à l'écran, devra être représenté d'une façon différente selon le type et devra se déplacer selon les exigences énoncées plus bas.

- Les classes à utiliser pour la partie 1 sont présentées dans le schéma en annexe.
 - o La position du véhicule est indiquée par la classe Position.
 - o La vitesse est exprimée en pixels.
 - o L'immatriculation est une chaîne de caractères.
- **L'immatriculation de chaque véhicule doit obligatoirement débiter par une lettre représentative de la classe. Vous devez procéder à cette vérification.**
- **La vitesse des véhicules de promenade doit être de 1 ou 2. La vitesse des véhicules d'urgence doit être de 1 ou 2 ou 3. Vous devez faire cette vérification.**
- **Chaque type de véhicule doit être représenté par un symbole particulier déterminé par la classe.**
- Tous les véhicules, sauf le nouveau type à ajouter, doivent se déplacer en utilisant un itinéraire répondant à l'équation de la droite : $y = ax + b$ (voir détails plus loin). Le déplacement doit aussi respecter les règles énoncées ci-après.
- Description des types de véhicules :
 - o VehiculePromenade : représente un véhicule standard. Ce type de véhicule se déplace toujours selon la vitesse indiquée.
 - o Ambulance : Une ambulance doit répondre à des urgences, elle doit donc se déplacer plus rapidement. Pour cette raison, les ambulances se déplacent toujours 3 fois plus rapidement que la vitesse indiquée.

- Pompier : Un véhicule de pompier aussi doit répondre à des urgences. Pour cette raison, ce type de véhicule se déplace toujours 2 fois plus rapidement que la vitesse indiquée.
 - **Autre véhicule : Ajoutez un autre type de véhicule, que vous nommerez comme vous voulez et qui apparaîtra à l'endroit où vous voulez dans la hiérarchie des véhicules. Ce type de véhicule a une façon particulière de se déplacer : il se déplace en x jusqu'à avoir atteint le x désiré, pour ensuite se déplacer seulement en y.** 
 - Le projet contient déjà une partie des classes respectant le schéma : Position, Vehicule et VehiculeUrgence.
 - **Vous devez évidemment ajouter des méthodes dans les classes mais vous ne devez pas ajouter d'attributs.**
 - **Les classe Vehicule et VehiculeUrgence doivent être abstraites. Vous devez indiquer les méthodes qui doivent être virtuelles pures dans ces classes.**
 - Les classes que vous ajoutez doivent comporter un constructeur par défaut ainsi qu'un constructeur par recopie.
 - Une partie de la classe Donnee aussi est fournie.
 - Elle contient un tableau de 50 véhicules.
 - Certaines méthodes existent déjà. Vous pouvez en ajouter d'autres.
 - Le constructeur par défaut de Donnees appelle InitialiserFlotteDepart() qui appelle AssignerValeurs(). Complétez le code d'AssignerValeurs() afin de donner les bonnes valeurs au véhicule reçu en paramètre. Dans le cas où les valeurs reçues ne sont pas valides, la fonction doit retourner false. Faites aussi la fonction AjouterVehicule() qui doit ajouter le véhicule à la 1^{ière} position vide de la flotte de véhicules.
 - Au début de la 'form' est déclaré un objet de type Donnees. Ainsi, le constructeur de Donnee est exécuté et les 2 véhicules de départ sont créés et ajoutés à la flotte. Une variable permettant de manipuler le véhicule courant aussi est déclarée. Vous ne devez pas avoir d'autres variables globales.
 - En tout temps, les véhicules doivent être affichés au bon endroit à l'écran à l'aide du bon symbole. Vous devez compléter la fonction AfficherUnVehicule().
 - Les boutons permettant de créer de nouveaux véhicules sont tous reliés à la même fonction. Cette fonction ouvre une nouvelle fenêtre afin d'indiquer les valeurs à utiliser pour créer le nouveau véhicule. Complétez cette fonction afin de créer le bon type de véhicule et de l'ajouter à la flotte de véhicules. Le véhicule s'affichera à l'écran au bon endroit.
- Note : Testez le sender pour savoir quel bouton a été utilisé.
- Le bouton Déplacer doit permettre de déplacer le véhicule dont l'immatriculation est sélectionnée dans le comboBox. Le bouton ouvre une nouvelle fenêtre afin d'afficher les données actuelles du véhicule choisi et pour permettre d'entrer la position où le véhicule doit se déplacer. Suite à la fermeture de la fenêtre, on doit voir le véhicule se déplacer à l'écran selon l'équation de la droite (sauf le nouveau type).
 - Faites varier le x en utilisant la vitesse du type de véhicule et calculer le y correspondant.
 - Il faut ajouter des règles particulières lorsque le x destination < x courant

- Dans le cas où le x destination est le même que le x courant, le véhicule ne se déplacera pas.
- Utilisez round() pour calculer le y.
- La boucle pour déplacer se trouve dans le programme principal. Les classes ne font que modifier les coordonnées et vérifier si la destination est atteinte.
- Finalement, le bouton Enlever Véhicule doit enlever le véhicule sélectionné de la flotte des véhicules. Il faut donc détruire le véhicule en question. N'oubliez pas de remettre le pointeur à NULL dans le tableau pour pouvoir utiliser l'emplacement de nouveau lors d'un prochain ajout. L'instruction pour faire disparaître le label de l'écran est déjà présente dans le code.
- Des messages d'erreur appropriés doivent s'afficher lorsqu'une action (ajout, déplacement, effacement) ne peut être effectuée.

Critères de correction et répartition des points :

- Respect du schéma de classes
- Application des règles de la programmation objet
- Respect du standard de programmation et des bonnes pratiques de programmation
- Utilisation adéquate des classes et des objets
- Fonctionnement selon les règles énoncées
- Respect des exigences demandées
- Respect des pratiques en orienté objet
- Répartition des points de cette partie :
 - Programmation des classes.....**15 pts**
 - Créer la flotte de départ.....**18 pts**
 - Afficher un véhicule**18 pts**
 - Créer un nouveau véhicule**14 pts**
 - Déplacer un véhicule.....**25 pts**
 - Enlever un véhicule**10 pts**

Schéma de classes

